Online offline api

The online/offline API

The online/offline API is used to detect changes in the networks status as well as checking the current status. The API broadcast two events: `window.ononline` *and* `window.onoffline`*. The network status can be checked using* `window.navigator.onLine`*.*
Disabling uneccessary calls to the network when it is not available or disallowed reduces cost, battery consumption and brings stability to the application.
To see priciples for good miniview implementation see Handling online/offline in WRT widget

window.navigator.onLine

`window.navigator.onLine` is a property that holds the current network state. In the HTML 5 working draft, this is either true or false, depending on whether or not the network is available. However, in the WRT platform this is extended with new Nokia-specific constants to three possible values:

- 0. `window.navigator.NetworkNotAllowed`
- 1. `window.navigator.NetworkAccessAllowed`
- 2. `window.navigator.NetworkAccessible`

The value of `navigator.onLine` is set to `navigator.NetworkNotAllowed (0)` when any of the following conditions are met:

- AllowNetwork access is not set in the info.plist;
- The widget is on the home screen, and the home screen is set to offline mode;
- The user chooses not to give permission to connect to the network.

The value of `navigator.onLine` is set to `navigator.NetworkAccessAllowed (1)` when any of the following conditions are met:

- AllowNetwork access is set to true in the info.plist;
- The widget is on the home screen, and the home screen is set to online mode;
- The user chooses to give permission to connect to the network or the network dialog is still waiting for the user selection (this widget tries to create a new connection).

The value of `navigator.onLine` is set to `navigator.NetworkAccessible (2)` when all the conditions for NetworkAccessAllowed are met and an active connection already exists.

```
var onlineMode = window.navigator.onLine;
alert(onlineMode);
```

Note: window.navigator.onLine should be checked in startup to see the initial network state. window.online or window.offline are only fired when the connectivity status changes.

window.onoffline

The `window.onoffline` event is fired whenever the value of `navigator.onLine` changes to `navigator.NetworkNotAllowed (0)`. This is a simple event meaning that it does not bubble and it cannot be canceled.

```
window.onoffline =  function(){
//stop periodic callbacks to network
//show offline indicator on mini-view
}
```

window.ononline

The `window.ononline` event is fired every time the value of `navigator.onLine>` is changed to either:

- 1: `window.navigator.NetworkAccessAllowed` or
- 2: `window.navigator.NetworkAccessible.`

The current online network state is passed as an argument to the event listener, which is bound to the `window.ononline` event.

This is a simple event, meaning that it does not bubble and it cannot be canceled. In both cases, the widget can try to start fetching updates from the network.

Example:

```
window.ononline = function(onOnlineState){
```

```
    //onOnline can be navigator.NetworkAccessAllowed  (1) or navigator.NetworkAccessible  (2)
    //perform periodic update if needed
    //hide the offline indicator from the miniview
}
```

Note: Connecting to the network when `navigator.onLine` is in the `NetworkAccessAllowed` state will cause an Internet access point dialog to show, unless a default access point is configured in the browser settings.

Code Snippet

This code snippet demonstrates how to add a online / offline functionality to WRT widget. The resulting program shows a status icon and outputs debug messages:

You can download the widget using the code snippets from Media:Using_online_offline_api_in_wrt_widget.zip, just rename it as .wgz to be able to it install it.

The widget demonstrates the online/offline API. There are **ononline** and **onoffline** event handlers added to the window object, and the data fetch request functions are added online/offline API functionality.

The Online/Offline API works only in specific devices and software versions, see Homescreen widget guidelines for details. The widget was tested to work in Nokia N97 v 20.0.019 with web browser version BrowserNG/7.1.18124.

Source: Relevant HTML components

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <link rel="StyleSheet" href="style/general.css" type="text/css" />
        <!-- script including the debug(message) function -->
        <script type="text/javascript" src="script/Debug.js" charset="utf-8"></script>
        <!-- script defining the Ajax object -->
        <script type="text/javascript" src="script/AjaxThing.js" charset="utf-8"></script>
        <script type="text/javascript" src="script/common.js" charset="utf-8"></script>
        <!-- main javascript file -->
        <script type="text/javascript" src="script/wrtstub.js" charset="utf-8"></script>
        <title>WRT Stub</title>
    </head>
    <body>
        <div id="bodyContent" class="bodyContent">
         <p class="status-info">Status icon: <span id="statusIcon" class="status-icon"></span></p>
        <!-- There will be debug messages added in the end of this div -->
        </div>
    </body>
</html>
```

Source: JavaScript

The `window.ononline` function is called if the window.navigator.onLine function changes to window.navigator.NetworkAccessAllowed or window.navigator.NetworkAccessible.

The `window.onoffline` function is called if the window.navigator.onLine function changes to window.navigator.NetworkNotAllowed.

Notice that the connectionStatus == CONNECTION_STATUS_OK corresponds to at least two different status icons, namely the "normal" icon and the "active" icon, corresponding to the states where the network access is allowed and where the network is accessible (the data connection is established), respectively.

The `window.navigator.onLine` informs the online status and can have three values:

onLine constant values

Constant	Value	Meaning
window.navigator.NetworkNotAllowed	0	Network is not allowed for this widget
window.navigator.NetworkAccessAllowed	1	Network is allowed for this widget
window.navigator.NetworkAccessible	2	There is an active data connection established and network is allowed

```
// Other than OK status overrides the default status obtained from
// window.navigator.onLine
var CONNECTION_STATUS_UNUSED = -1;
var CONNECTION_STATUS_UNINITIALIZED = 0;
var CONNECTION_STATUS_REQUEST_INITED = 1; // When the connection is initialised
var CONNECTION_STATUS_CONNECTING = 2; // When the connection is sent
var CONNECTION_STATUS_OK = 3; // Use the window.navigator.onLine to
                              // decide the status
var CONNECTION_STATUS_FAILED = 4; // When the connection request fails
var CONNECTION_STATUS_NOT_ALLOWED = 5; // When the connection is disabled
var CONNECTION_STATUS_HANDLING_ERROR = 6; // Parsing errors etc.

// Initializes ononline and onoffline functions to change the status if needed
function initOnonlineAndOnoffline() {
    window.onoffline = function(){
        if(requestObject == null) {
            // update the status only if there is no existing fetch request
            var newConnectionStatus = CONNECTION_STATUS_NOT_ALLOWED;
            updateStatusInfo(newConnectionStatus);
        }
        else {
            // else the existing fetch request updates the status
            debug('Set to Offline, request is in progress');
```

```
        }
    };
    window.ononline = function() {
        if(requestObject == null) {
            debug('Set to Online, updating status');
            // update the status only if there is no existing fetch request
            var newConnectionStatus = CONNECTION_STATUS_OK;
            updateStatusInfo(newConnectionStatus);
        }
        else {
            // else the existing fetch request updates the status
            debug('Set to Online, request is in progress');
        }
    }
}


// Processes request
function sendRequest() {
    debug('Sending request');

    // check if the fetchURL is valid and there's no update currently
    // in progress
    if ((fetchURL != null)
        && (requestObject == null)) {
        // Shortcut to return if in offline mode
        if (window.widget && (typeof window.navigator.onLine=='number')) {
            if (window.navigator.onLine == window.navigator.NetworkNotAllowed)
            {
                var newConnectionStatus = CONNECTION_STATUS_NOT_ALLOWED;
                updateStatusInfo(newConnectionStatus);
                releaseResources();
                debug("Widget is offline, can not send the request!");
                return;
            }
        }

        // show progress information if this is a commanded feed update
        if (showDetails) {
            var newConnectionStatus = CONNECTION_STATUS_CONNECTING;
            updateStatusInfo(newConnectionStatus);
        }

        // fetch the specified URL
        requestObject = new AjaxThing();
        var url = fetchURL;
        // Note that each url will be different, thus they are not cached.
        url=url+"&sid="+Math.random();
        requestObject.onreadystatechange=stateChanged;
        requestObject.open("GET",url,true);
        requestObject.send(null);

        debug("Request sent...");
    }
}

// Called when the asynchronous ajax response is obtained
function stateChanged() {
    if (requestObject.readyState == 4) {
        var responseStatus = null;
        try {
            responseStatus = requestObject.status
        }
        catch (e) {
        }
        fetchCompleted(responseStatus);
    }
}


// Callback function that gets called when the fetching has completed.
function fetchCompleted(responseStatus) {
    var newConnectionStatus = CONNECTION_STATUS_UNUSED;

    if (responseStatus == 200) {
        newConnectionStatus = CONNECTION_STATUS_OK;

        /* Here: get the response from the requestObject, handle it and
         * possibly set the newConnectionStatus to be some failure state
         * like newConnectionStatus = CONNECTION_STATUS_HANDLING_ERROR;
         */

        debug("Response was handled.");
    } else {
        // We could notice here that the connection is disabled, i.e.,
        // the short cut of sendRequest is not working when the user
        // denies the connection when asked for the first time.
        if (window.widget && (typeof window.navigator.onLine == 'number')) {
            switch (window.navigator.onLine) {
                case window.navigator.NetworkNotAllowed:
                {
                    newConnectionStatus = CONNECTION_STATUS_NOT_ALLOWED;
                    break;
                }
                // If the network access is allowed, there could be another
                // reason for the error, e.g., wrong URL address.
                case window.navigator.NetworkAccessAllowed:
                case window.navigator.NetworkAccessible:
                {
                    newConnectionStatus = CONNECTION_STATUS_FAILED;
                    break;
                }
            }
        }
        else {
            newConnectionStatus = CONNECTION_STATUS_FAILED;
        }
        debug("Error in the connection!");
    }

    // Now the response is handled.
    // Update the status and release the resources
    updateStatusInfo(newConnectionStatus);
    releaseResources();
}

// Called after the current request process has ended.
function releaseResources() {
    // null the requestObject reference to indicate that there's no current
    // request in progress
```

```
    requestObject = null;
}

// Updates the status icon
function updateStatusInfo(newConnectionStatus) {
    connectionStatus = newConnectionStatus;
    var spanClass = '';
    switch (connectionStatus) {
        case CONNECTION_STATUS_OK:{
            if (window.widget
                && (typeof window.navigator.onLine == 'number')) {
                switch (window.navigator.onLine) {
                    case window.navigator.NetworkAccessAllowed:
                    {
                        spanClass = 'normal';
                        break;
                    }
                    case window.navigator.NetworkAccessible:
                    {
                        spanClass = 'active';
                        break;
                    }
                    case window.navigator.NetworkNotAllowed:
                    {
                        spanClass = 'disabled';
                        break;
                    }
                    default:
                    {
                        spanClass = '';
                    }
                }
            }
            else {
                spanClass = 'active';
            }
            break;
        }
        case CONNECTION_STATUS_CONNECTING: {
            spanClass = 'connecting';
            break;
        }
        case CONNECTION_STATUS_REQUEST_INITED:{
            spanClass = 'waiting';
            break;
        }
        case CONNECTION_STATUS_HANDLING_ERROR:{
            spanClass = 'error-handling';
            break;
        }
        case CONNECTION_STATUS_FAILED:{
            spanClass = 'error-failed';
            break;
        }
        case CONNECTION_STATUS_NOT_ALLOWED: {
            spanClass = 'disabled';
            break;
        }
        default : {
            spanClass = '';
        }
    }
    var i;
    debug('status="'+spanClass+ '", connectionStatus:'+connectionStatus
          + ', onLine:'+window.navigator.onLine);
    spanClass = 'status-icon '+spanClass;
    document.getElementById('statusIcon').setAttribute('class', spanClass);
}
```

The `requestObject` is a XMLHttpRequest object, which is not null when there is an active request in progress.

The `updateStatusInfo` function updates the status icon by updating the class of the #statusIcon span element to correspond to the connectionStatus.

Supplementary material

This code snippet is part of the stub concept, which means that it has been patched on top of a template application in order to be more useful for developers. The version of the WRT stub application used as a template in this snippet is v1.2.

- The patched, executable application that can be used to test the features described in this snippet is available for download at File:Using online offline api in wrt widget.zip.
- For unpatched stub applications, see Example stub.